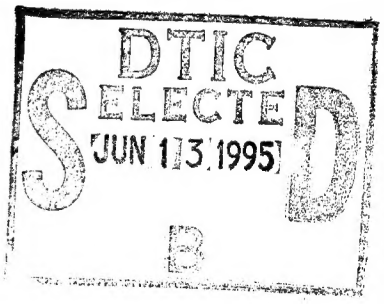
RL-TR-95-41
Final Technical Report
March 1995

A METHODOLOGY FOR APPLICATION DESIGN USING ACTIVE DATABASE TECHNOLOGY

Georgia Institute of Technology

Shamkant Navathe, Asterio Tanaka, Ramesh Madhavan,
and Yee Huat Gan



Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-95-41 has been reviewed and is approved for publication.

APPROVED: Raymond A. Liuzzi

RAYMOND A. LIUZZI
Project Engineer

FOR THE COMMANDER: Henry J. Bush

HENRY J. BUSH
Deputy for Advanced Programs
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (C3CA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1995	Final Jun 93 - Oct 94

4. TITLE AND SUBTITLE
A METHODOLOGY FOR APPLICATION DESIGN USING ACTIVE DATABASE TECHNOLOGY

5. FUNDING NUMBERS
C - F30602-93-C-0175
PE - 62232N
PR - R427
TA - 00
WU - P4

6. AUTHOR(S)
Shamkant Navathe, Asterio Tanaka, Ramesh Madhavan, and Yee Huat Gan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Georgia Tech Research Corporation
Centennial Research Building, Room 246
Georgia Institute of Technology
Atlanta GA 30332-0420

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NCCOSC RDTE Div
53560 Hull St
San Diego CA 92152-5001

Rome Laboratory (C3CA)
525 Brooks Rd
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-95-41

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Raymond A. Liuzzi/C3CA/(315) 330-3528

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
Even though there has been a fair amount of work done in the incorporation of active behavior into database systems, little research has addressed the conceptual, DBMS-independent representation of the active behavioral aspects of the applications. This report presents the work done in the introduction of active aspects of information systems at the conceptual schema level. ER is described, which is an extension of the ER model to express active database behavior in the form of events and rules. Also described is a language to accommodate the concepts of events and rules in the ER schema. A methodology is proposed for the design of active databases based on the ER model, as an extension for traditional database design methodology. With the extension, the data model mapping step includes the translation of active database behavior specified in the conceptual schema into triggers or event alerters written in the DBMSs DDL/DML. Finally, the implementation of a graphical tool is described, which can do the mapping from the ER model to the logical schema of the target database.

14. SUBJECT TERMS
Computers, Software, Database, Artificial intelligence

15. NUMBER OF PAGES
28

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

A Methodology for Application Design using Active Database Technology : Final Report

Shamkant Navathe - P.I.
Asterio Tanaka
Ramesh Madhavan
Yee Huat Gan

Abstract

Even though there has been a fair amount of work done in the incorporation of active behavior into database systems, little research has addressed the conceptual, DBMS-independent representation of the active behavioral aspects of the applications. This report presents the work done in the introduction of active aspects of information systems at the conceptual schema level. We describe (ER)^2 ; an extension of ER model to express active database behavior in the form of events and rules. We also describe a language to accommodate the concepts of events and rules in the ER schema. We propose a methodology for the design of active databases based on the (ER)^2 model, as an extension for traditional database design methodology. With this extension, the data model mapping step includes the translation of active database behavior specified in the conceptual schema into triggers or event alerters written in the DBMS's DDL/DML. We also describe the implementation of a graphical tool which can do the mapping from the (ER)^2 model to the logical schema of the target database.

1 Introduction

Database systems have evolved and become very popular in the past few decades, and their scope of application has expanded from traditional business data processing systems to new classes of non-conventional information systems. There has been an increasing interest in providing new functionalities to DBMSs, so that they can fully support such non-conventional applications. Active database capability is one such functionality of the next-generation DBMSs.

Traditional DBMSs are passive in that they only execute explicit requests from user queries or application programs. At best, they may trigger update actions to enforce referential integrity constraints. Such limitation complicates the use of traditional DBMSs in

information systems that deal with problems concerned with observation of tasks and situations such as: air traffic control, military operations planning, process control/fault diagnosis network management, hospital monitoring systems, inventory control, economic surveillance and forecasting, and cooperative problem solving, to name a few. These applications typically require monitoring the database state (both static and transitional) and reacting to certain events.

An active DBMS has been defined as a system with full database functionality and the additional capability of monitoring the state of the database and executing some predefined actions when appropriate events are detected. An active DBMS must have this functionality plus the ability to react independent of an external request. Thus, an active database behavior can be defined as the enforcement of database constraints and situation/action rules through actions initiated by the database system when the appropriate events occur, independent of external requests.

1.1 Problem Overview

Current design methodology for relational databases comprises a conceptual design phase, where the relevant part of the real world is modeled in terms of a conceptual schema representing the objects of interest, their properties, and the inter-object relationships. The Entity-Relationship (ER) model is the "de-facto" standard conceptual model used in the design of relational databases. A conceptual schema defined in the ER model contains both structural and semantic information and provides DBMS independence.

When the active database behavior is incorporated into the DBMSs, there are no corresponding constructs in the conceptual model that can capture the kind of functionality represented by the incorporation of general-purpose database triggers and event alerters. In addition, while production rules and stored procedures are very powerful language constructs capable of modeling any computable database triggers or event alerters, an inherent complexity is introduced when one has to incorporate them into programs. The reason is that they are procedural and unstructured, in contrast to the declarative nature of the DDL/DMLs. A number of practical problems arise in the design and implementation of database triggers and event alerters using rules and procedures.

- Rules are difficult to understand and control: they can fire indefinitely and may go into infinite loops; conflicts are possible, and hard to predict.
- The enforcement of complex integrity constraints and business rules usually requires the coding of several database triggers to cover all potential invalidating events.
- There are significant differences between the language constructs in different DBMSs.

As a consequence, designing the structure and behavioral aspects of a database that fully incorporate the active behavior of a given set of applications using production rules and stored procedures can be a very difficult task.

1.2 Motivation

Even though there has been a fair amount of work done in the incorporation of events, rules, and triggering mechanisms into database systems, little research has addressed the conceptual, DBMS-independent representation of the active behavioral aspects of the applications. In addition, with the enlargement of the gap in the database design process due to the inclusion of production rules and stored procedures in the DDLs/DMLs, new tools are required to specify and translate active database behavior from the conceptual to the DBMS level, as well as to analyze and validate the specification of behavior. Such tools will help in overcoming the inherent complexity of programming with these language constructs.

This paper reports the work done in the introduction of active aspects of information systems in the conceptual schema. We describe $(ER)^2$: an extension of the ER model to express active database behavior in the form of events and rules. This extension allows the capturing of more application semantics at the conceptual level, along an axis orthogonal to static data modeling and process modeling. We also describe a language to accommodate the concepts of events and rules in the ER schema. We propose a methodology for the design of active databases based on the $(ER)^2$ model, as an extension to the traditional database design methodology. With this extension, the data model mapping step includes the translation of active database behavior specified in the conceptual schema into triggers or event alerters written in the DBMS's DDL/DML. We also describe the implementation of a graphical tool which can do the mapping from the $(ER)^2$ model to the logical schema of the target database.

The following advantages can be derived from the extended modeling and design methodology.

- Reduced database design effort : the burden of capturing the behavior of the model will be shifted to the tool.
- Reduced application development effort : it is accomplished by an automatic translation of events and rules into DBMS language constructs.
- Better control of application development : this accrues from introducing precision in the specification of behavior.
- Better quality of overall design of the database and the applications : this is realized by an interactive development environment capturing the knowledge of experts and automatically dealing with the constraints of the DBMS.

The rest of the report is organized as follows. The next section describes the overall approach to the application design of active databases. It includes the present application design approach in traditional databases, a description of the $(ER)^2$ model and the design approach using the $(ER)^2$ model. Section 3 gives a description of the active database design tool.

2 Overall Approach to Active Database Design

This section describes the database design methodology. The initial subsection gives an overview of the current state of practice of relational database design using the well established ER approach. The next subsection introduces the $(ER)^2$ model which is an extension of the ER model to express active database behavior in the form of events and rules. The third subsection describes the database design approach for active databases using the $(ER)^2$ model.

2.1 Application design using ER model

The conceptual level is an important part in the three level ANSI architecture. A conceptual schema can be defined as the total information contents of the database, both of its structure and the semantics. The conceptual schema is the determinant to the concept of data independence.

The most widely used semantic data model is the ER model and its variations. The basic ER model does not allow the expression of the semantics of the real world. A number of extensions and variations have been proposed to express additional semantics [1]. The ER model can be represented graphically by an ER diagram.

The process of mapping the data model from ER to relational in the relational database design methodology has been mentioned in section 1.2. This is a very well understood process (e.g., see [3]), strongly supported in theory and largely used in practice. The mapping takes place in two stages. In the first stage, an abstract relational schema is generated from the ER schema that is independent of any specific features of the DBMS implementation of the relational model. In the next stage the abstract relational schema is mapped into a collection of data definition statements in the language of a specific DBMS.

2.2 The $(ER)^2$ Model

The ER approach uses a set of simple concepts and a graphical formalism that is easy to understand. However, the major drawback of the ER approach is the lack of constructs for specifying behavior. Domain constraints such as "salary cannot be less than 10,000 or greater than 100,000" cannot be modeled in the ER model, unless it is possible to define a range of permitted values for an attribute type. Constraints involving derived information, e.g. "the total weight of the passengers on the plane cannot exceed 20 tons", also cannot be modeled in the ER diagram. Dynamic constraints on state transitions, or constraints involving temporal aspects cannot be enforced in the basic ER formalism. Most of these constraints are currently described in application programs as validation rules. Because many programs use the same data, rules must be repeated in every program, thus creating redundancy and, as a consequence, a source of inconsistency.

Recent research efforts have tried to overcome the above mentioned drawbacks. There are two classes of solutions: basic approaches, in which integrity constraints are formulated

in combination with the ER model, without introducing fundamentally new concepts, and novel approaches, in which the concepts of events and actions are introduced, thus adding new constructs to the ER model.

The basic approach could be either static or dynamic. In the static approach, all rules are defined directly as a set of structural integrity constraints which must hold for each state of the database. For dynamic rules, either a single state or a pair of consecutive states of the database are involved. This means that the database contains not only the current state of the universe of discourse, but also all relevant historical information, while the ER diagram must describe this mixture of current and historical information. External events are modeled as relationships in the ER diagram, and help to establish the link between two successive states. The drawback is the need for a large number of rules to describe all the interconnections between the different types of information amalgamated into a single diagram, making it unmanageable in most cases.

In the dynamic approach, all allowed and necessary update operations on the database are described. For this purpose, a general syntax is proposed to define constraints on the basic update operations. Instead of determining only whether a new database state is valid, it is possible to determine whether an update operation, which is involved in the creation of a new state is permissible. A number of drawbacks are inherently associated with the update oriented approach: rules may conflict, lead to an ambiguous situation, or lack sufficient flexibility; sometimes they can be checked only afterwards in an expensive way; historical information is projected in the current database state, thus adding to the complexity.

The novel approaches introduce new primitive concepts of action or event as a basis for conceptual design. By using actions as primitive constructs, the history of the system is reflected in the sequence of actions, so that it is no longer necessary to model all relevant past information in the current state of the database. But the novel approaches aim at transformation of the high level specification of the problem into an implementation model in a target programming language. These approaches lack most of the characteristics that have made database systems so successful. With the incorporation of dynamic features in the new generation of DBMSs, the trend is to avoid non-database solutions to database problems. In this sense, there is still room for extensions to the ER methodology for database modeling and design.

As a basis to the proposed extensions to the ER model, we adopt a variant of the ER model that includes generalization/specialization, both homogeneous and heterogeneous, and full aggregation which allows relationships involving relationships, thus requiring directed arcs in the ER diagram to denote inter object connections. In our approach, we view the real world as constituted by entities, relationships, events and rules, all primitive objects of the model. While entities and relationships, along with their attributes, represent the structural aspects of the information system being modeled, events and rules represent the active behavior that controls the states of the data objects and their attributes. We call the resulting model as the Entity Relationship model with Events and Rules, or $(ER)^2$ model for short. The grammar for the extensions are given in the appendix.

Figure 1 shows the meta-diagram of the $(ER)^2$ model. The two new objects "EVENT"

and "RULE" are incorporated into the ER model as specializations of "ER2_OBJECT". Like entities and relationships, events and rules may have attributes that describe their properties. Being first-class objects, they may be connected to each other and to other objects through meta-relationship "ER2_Connections". Possible connections involving events and rules are:

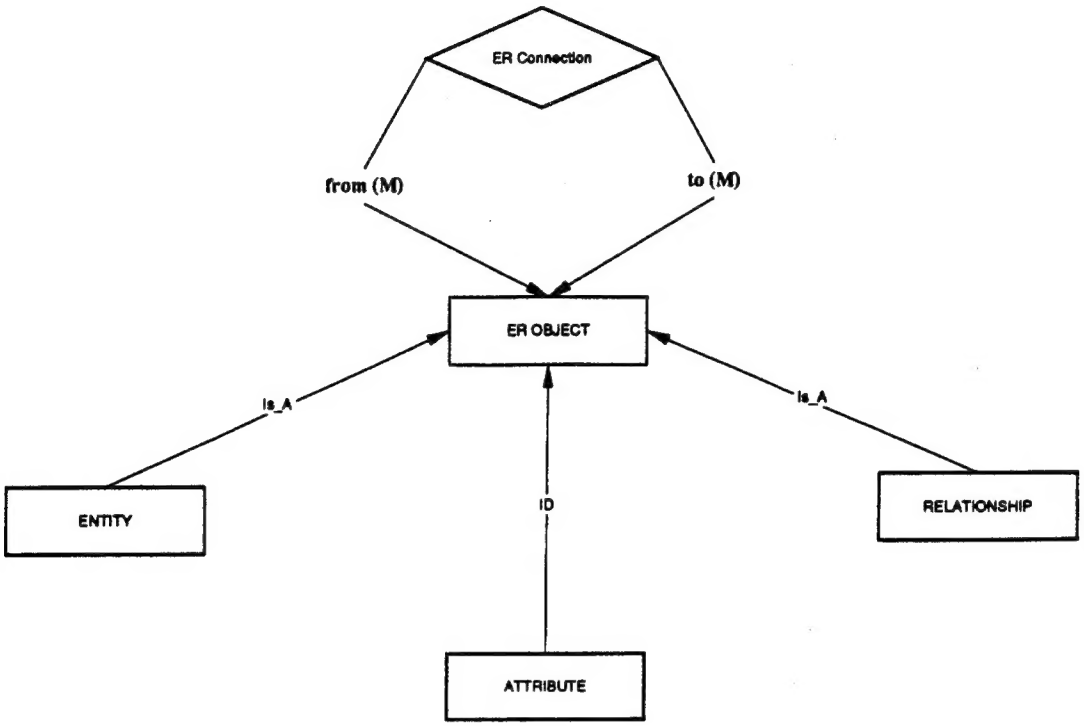


Figure 1: Meta-ER-diagram of the ER Model

- Inter-event connections: "Precedes" referring to the order of occurrence of events.
- Inter-rule connections: "Priority" to determine the order of execution of rules.
- Connections between events and rules: "Fires" between an event and the associated rules. "Raises" between a rule and the events that its execution gives rise to.
- Connections between events and entities/relationships: "Affects" meaning that an event raised by the execution of a rule may affect the state of an entity/relationship. "Affected-by" meaning that an entity/relationship is affected by the occurrence of an event.

2.2.1 Events

An event is something that happens at a point in time, and, theoretically has no duration. Events may logically precede or follow one another or may be unrelated. There are two types or ordering of events to be considered: a causal ordering and a temporal ordering. The first

one relates events of different types (e.g. the event "flight X landed" cannot occur before the event "flight X takes off"). Not all pairs of events bear this relationship. Causally unrelated events are said to be concurrent and can occur in any order because they have no effect on each other. The temporal ordering, on the other hand, is based on the linear ordering of the time of occurrence of the events. As far as the activation of the behavior is concerned, we consider that the time of detection of an event is the same as its time of occurrence. This may not be true for actual systems, but it is not a problem in conceptual modeling as long as the order is preserved [5].

We distinguish events that occur on data objects or attributes stored in the database (database events) from those events that are external to the database, usually generated by application programs (external events). The third type of events (system events) are signals generated by the underlying system such as interrupts and clock events.

2.2.2 Rules

A rule consists of a condition part and an action part. The syntax for rules is given in the appendix. The condition part of the rule is a predicate over the state of the database. The condition part could be empty, a simple condition, or a complex condition. The condition acts as a guard on action list. If the condition fails, no action will be triggered and the rule execution will fail. An action list is a sequence of commands that can be database actions or external actions like raising an external event or sending a message.

2.2.3 Formal specification of the $(ER)^2$ Model

In order to derive an operational semantics for the language, i.e. the semantics of the language in terms of the execution of its operations in an abstract computing machine, a formal specification of the $(ER)^2$ model is required. In this section we will give the formal specification of an event and the operational semantics of the action "raise". The complete formal specification for the $(ER)^2$ model can be found in [4].

An *event type descriptor* is a 4-tuple:

$$(event\_id, event\_attr\_set, event\_spec, fired\_rule\_set)$$

where

- $event\_id \in E$ is the identification of the event type;
- $event\_attr\_set \subset A$ is the (possibly empty) set of attributes carried by the event type;
- *event_spec* comprises either a database event in D or the name of the signal (external or system event) in S;
- $fired\_rule\_set \subset R$ is the (possibly empty) set of rules fired by the occurrence of the event type.

A *rule type descriptor* is a 7-tuple:

$$(rule\_id, rule\_attr\_set, priority\_level, firing\_event,$$

$$condition\_spec, action\_list\_spec, raised\_event\_set)$$

where

- $rule\_id \in R$ is the identification of the rule type;
- $rule\_attr\_set \subset A$ is the (possibly empty) set of attributes of the rule type (carried by its firing event);
- $priority\_level \in P$ is the identification of the priority level of the rule type, if any;
- $firing\_event\_set \subset E$ is the (at least singleton) set of event types that fire the rule;
- *condition_spec* is the specification of the condition part of the rule in the active behavior specification language;
- *action_list_spec* is the specification of the actions of the rule in the active behavior specification language;
- *raised_event_set* is the (possibly empty) set of event types raised by the execution of the rule.

RAISE $e : s(p\_l)$

Given *event_id* e, *signal_name* s, and *actual_parm_list* p_l, RAISE adds e to the list of detected events; s and the values in p_l must conform to the *event type descriptor* of e.

Pre-conditions:

1. $e \in E$
2. $s \in S$, $event\_name(e) = s$
3. If p_l is specified, let $p\_l =< a_1, a_2, ..., a_n >$
   and *event_attr_set* of $e =< A_1, A_2, ..., A_n >$. Then each value must be in the proper domain, i.e.,
   $a_i \in domain(A_i), 1 \leq i \leq n$

Execution:

Add e to the list of detected events.

Post-conditions:

e is raised.

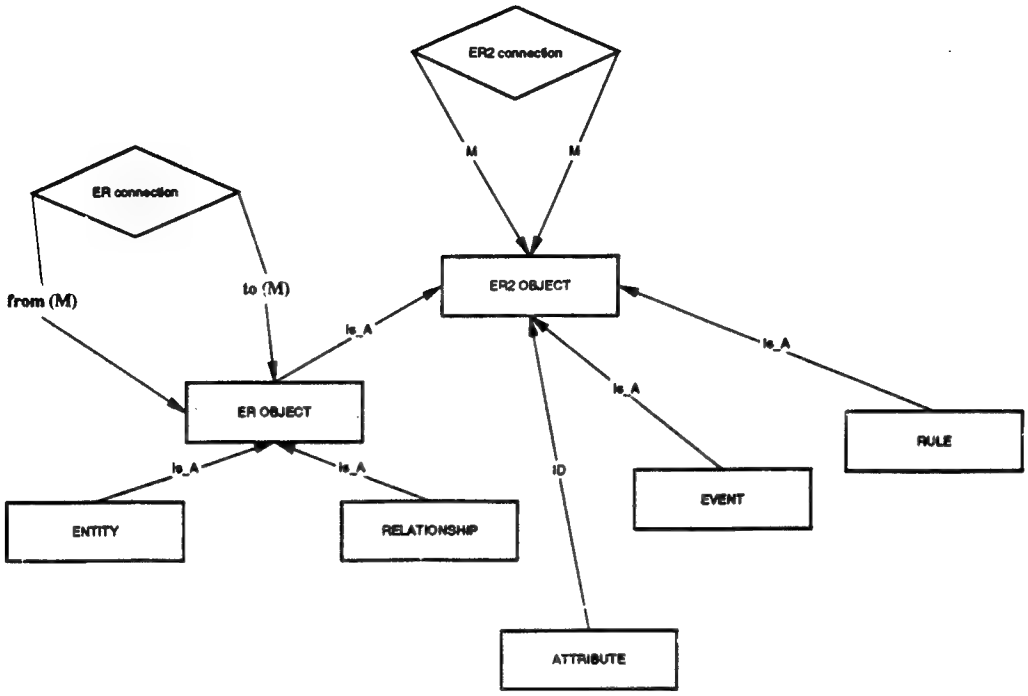


Figure 2: Meta-ER-diagram of the $(ER)^2$ Model

2.2.4 Meta-schema

The $(ER)^2$ model can be specified in a meta-schema and represented as an $(ER)^2$ diagram, which is a meta-$(ER)^2$-diagram of the $(ER)^2$ model itself. This is shown in Figure 2, where the event and rule objects are integrated in the model with the appropriate notation, and the "ER2_Connection"'s of Figure 1 are explicitly represented by the links "Affected_by", "Affects", "Fires", "Raises", and by the relationships "Precedes" and "Priority". The external environment (system , applications, and users) is also shown as a potential source and target of events.

2.2.5 Example

As an illustration, Figure 3 shows an $(ER)^2$ diagram of a company's EMPLOYEE-DEPARTMENT-PROJECT database with some events and rules attached to the data objects. The following ER schema is assumed - for simplicity, details such as cardinality ratios ("1","M"), identification dependencies ("ID"), participation constraints ("Total"), and roles ("manager", "employer") are shown only in the diagram, and attributes are specified only in the textual schema:

- EMPLOYEE (ssn, name, job, address, birth_date, status, salary)
- DEPARTMENT(name, location)

- PROJECT(<u>name</u>, budget)
- DEPENDENT(<u>EMPLOYEE_ssn</u>, <u>name</u>, birth_date)
- Employed(<u>EMPLOYEE_ssn</u>, <u>DEPARTMENT_name</u>)
- Manages(<u>EMPLOYEE_ssn</u>, <u>DEPARTMENT_name</u>)
- Works(<u>EMPLOYEE_ssn</u>, <u>PROJECT_name</u>, start_date, hours_week)

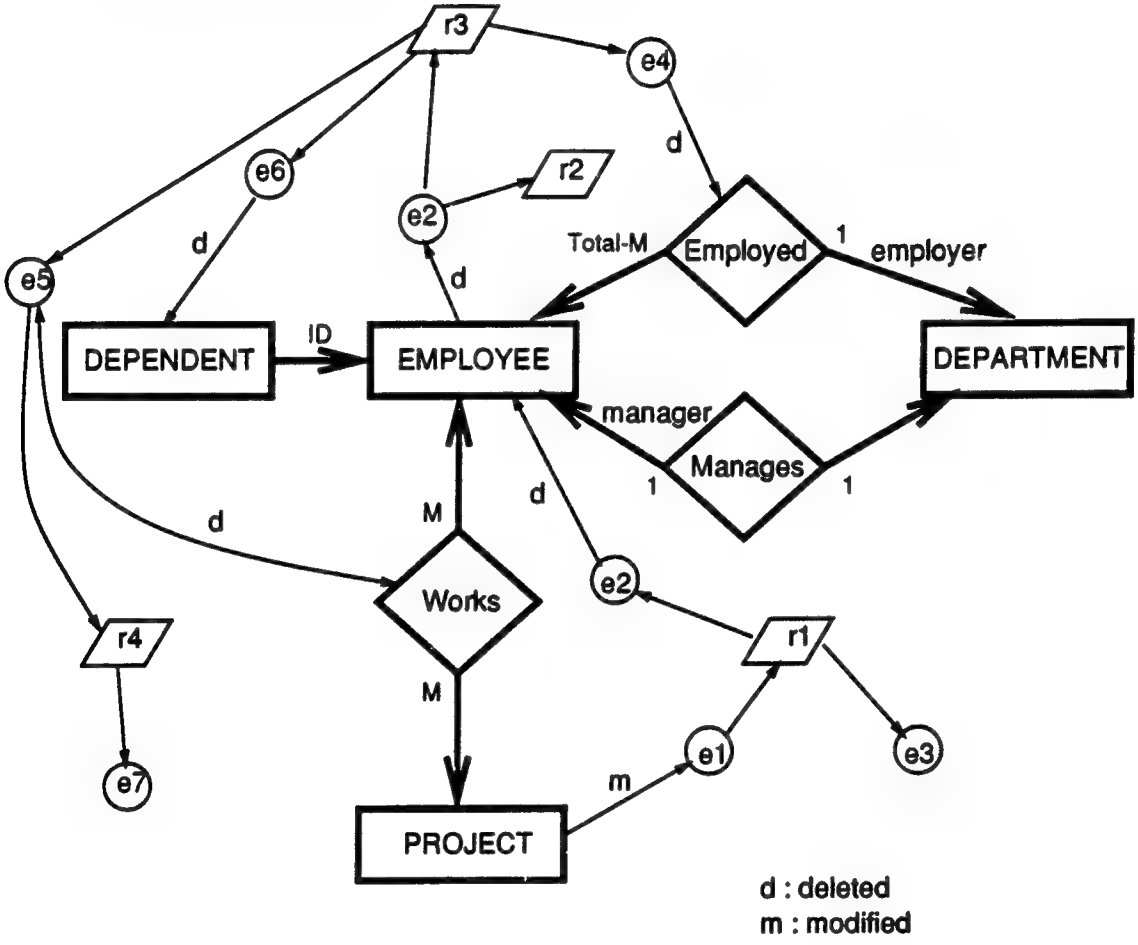


Figure 3: $(ER)^2$ Diagram of a COMPANY Database

The following behavioral sentences are specified in terms of the events and rules represented in Figure 3:

WHEN e1 : PROJECT MODIFIED
FIRE r1 ("Policy for budget reduction") :

IF NEW budget < OLD budget
THEN DELETE_ENTITY EMPLOYEE (ssn = OLD EMPLOYEE_ssn,
status = "temporary") (e2),
RAISE e3 : salary_review.

WHEN e2 : EMPLOYEE DELETED
FIRE r2 ("Restriction to firing engineers") :
IF OLD job = "engineer"
THEN MESSAGE : "Employee is an engineer, deletion rejected",
REJECT_OPERATION.

WHEN e2 : EMPLOYEE DELETED
FIRE r3 ("Cascaded deletion of temporary employees") :
IF OLD status = "temporary"
THEN PROPAGATE_OPERATION (e4 : Employed DELETED,
e5 : Works DELETED, e6 : DEPENDENT DELETED).

WHEN e5 : Works DELETED
FIRE r4 ("Warning message to project manager") :
MESSAGE : "Inform change on employee assignment to project manager"
(e7 : manager_warning).

The $(ER)^2$ diagram represents the active database behavior in the form of events and rules and their interaction with data objects. To avoid cluttering the diagrammatic representation, we chose to keep the specification of events, conditions, and actions apart from the diagram, using textual description. The same user interface design technique is adopted by most of the current ER diagramming tools, where the attributes are specified in pop-up windows that are displayed when the corresponding object symbols are clicked. This technique keeps the diagram simple and easy to read, without loss of information.

As shown in the above example, the $(ER)^2$ model can capture a variety of constraints and situation/action behaviors, such as a high-level organizational policy (rule r1), a restrictive prescription (rule r2), the enforcement of an integrity constraint (rule r3), or a database event alerter (rule r4). Potentially, this framework can represent any application-relevant behavior that can be managed by an active DBMS. In addition, this representation can be easily adapted to data abstraction extensions to the ER model such as generalization/specialization and aggregation. These extensions do not disturb the $(ER)^2$ framework because of the orthogonality of the added dimension.

2.3 Application design using $(ER)^2$ model

This section describes how an application can be developed using the ER^2 design methodology. The example schema shown in the previous section is used to demonstrate the design

methodology. The Ingres database management system is considered as the target database system for the application. In this section we only give the conversion steps that are required for the given example. The complete description of the exact steps to be followed in the conversion of the ER^2 diagram to the database specification can be found in [4].

The extraction of the description of the ER2-OBJECTS from the ER^2 diagram is pretty straightforward. This is the first step in the conversion and in this step the description of the entities, relationships, events, rules and their connections is extracted from the diagram. This can be used to get a database independent description of the schema and the specification of the behavior in the Abstract Behavior Specification language.

3 An Active Database Design Tool

Current database design methodology is supported by automated tools that perform the transformations and mappings during the design process. Since the modeling power of the conceptual schema is enhanced with the $(ER)^2$ model, it is a natural consequence to extend the set of tools that assist the design methodology. Among others, a clear benefit of these extensions is the accuracy of the executable definitions generated for both meta-behavior and application-related active behavior. In this section we describe the tool we are developing that can perform the transformations on a $(ER)^2$ diagram. The tool is being developed along the lines of ERDRAW developed at the Lawrence Berkeley Labs (LBL) which can do the transformations on ER diagrams.

3.1 Architecture of Design Tool

The proposed architecture of the tools is shown in Figure 4. The format of the intermediate files stored by the tool is similar to that used by ERDRAW. The ER2-GT tool allows the user to generate and modify an $(ER)^2$ diagram which can also store a table representation of the diagram. This will contain the table representation of the both the ER objects as well as the events and rules in the diagram. The former part is stored in the same way as done by ERDRAW and the latter part is stored in a similar format. The stored description of the ER objects can be converted to database specific data description using the Schema Definition Tool (SDT) from LBL. The latter part of the description can be converted by the Active Behavior Translation Tool to the active behavior specification language. The same description can be converted to the event specification language of any target database.

3.2 Capabilities of the tool

Since the amount of information contained in a sample application could be large, displaying all the information together in a single screen will be very confusing to the user. Hence the ER2-GT tool is basically structured into a hierarchy of layers. The base level is the same

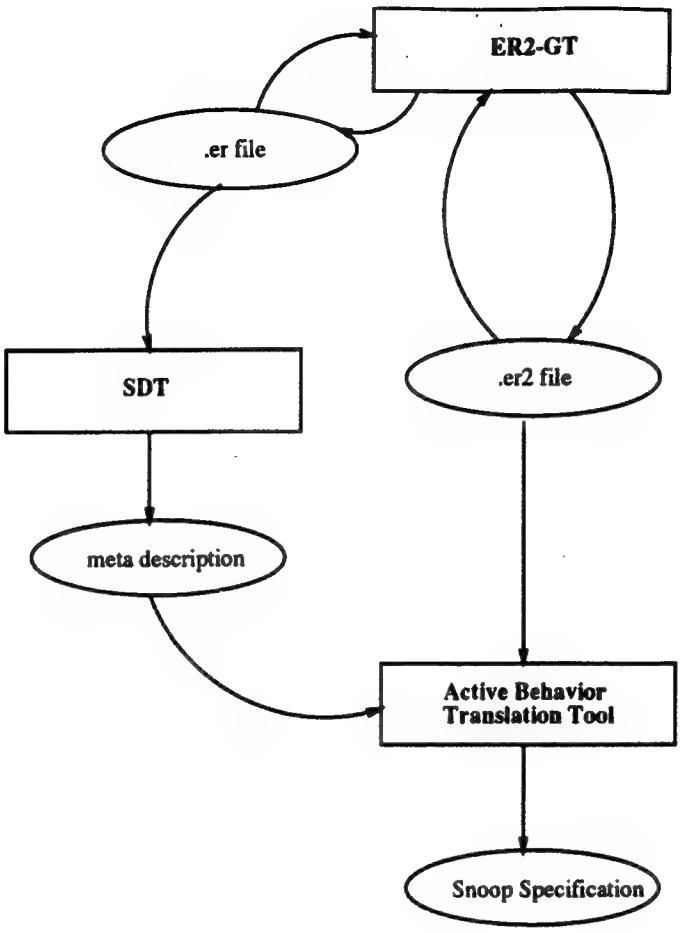


Figure 4: Architecture of the tool set

as the ERDRAW where the ER objects are displayed. In this layer only the entities, relationships and their interconnections are displayed. The attributes are not directly displayed in the diagram but they can be displayed textually on demand on a per object basis. In addition to the base ERDRAW display, in the ER2-GT tool, the behavioral relationship between object through an Event-Condition-Action (ECA) rule is also displayed.

The details of the behavioral relationships are shown in the next level of the hierarchy when one of them is selected from the base level display. This is displayed in a different window so that the user can have much more relevance to the context. This level will display the entities or relationships, the events, and the rules involved in the particular ECA rule. However, as in the base level, the details of the objects are not directly displayed in the diagram. The details can be displayed textually by selecting the object and inquiring its details.

In addition to the drawing capability, the tool is designed to have some additional capabilities like displaying the relevant information about an object. This could be something like inquiring about the rules that are affecting a given entity. Such facilities give more flexibility to the user in making good and correct design. Moreover, the tool allows the user to store

a given $(ER)^2$ diagram in a file and retrieve and display of an already stored diagram from a file.

The active behavior translation tool is capable of converting a file stored by the ER2-GT tool into an equivalent set of statements in the active behavior specification language. It is straightforward to extend this tool to be able to generate behavior specification in any target DBMS language.

Currently, the tool is designed for the Active Behavior Specification Language as described in [4]. We would like the tool to have the capability to generate the specifications in terms of Snoop [2], which is the Active Behavior Specification Language for Sentinel. However, as Snoop has a much richer active behavior specification than $(ER)^2$, we would need to perform a mapping between the graphical symbols represented in $(ER)^2$ to Snoop. For example, Snoop allows the specification of complex events which cannot be represented directly with an $(ER)^2$ diagram. Other Snoop events such as periodic and aperiodic events are also not represented. This task is not trivial as there is a trade-off between representing as much active behavior information as possible and cluttering the diagram with too many symbols. If too much information is shown in one diagram, the resulting picture will be too confusing to be of any use. More careful study needs to be performed to obtain the optimal design.

4 Conclusion

Our thesis was that the lack of modeling constructs for active database capabilities present in the new generation of relational DBMSs has made it difficult to take full advantage of their potential benefits. The current database design methodology forces the user to defer critical modeling decisions concerning the active behavior of the database to late stages of the design process, where the semantics of the real-world situations are obscured by the intricacies of the implementation model. Because of the inherent complexity of rule-based programming, database designers do not exploit adequately the functionalities of rules, triggers and stored procedures. Furthermore, it is expected that more powerful active capabilities will be added to the DBMSs by non-conventional database applications, enlarging the gap between modeling and specification of executable definitions of active behavior.

Our approach to this problem was to extend the well-established methodology based on the ER model by incorporating active database behavior in the form of events and rules as first-class objects of the model. Besides providing the modeling constructs, we proposed an extended architecture of tools for assisting the database designer in the task of specifying, analyzing, and translating active behavior into executable data definition statements in DBMSs.

The following benefits will result from the extended modeling and design methodology: reduced database design and application development effort with the automatic generation of meta-behavior and translation of active behavior into executable DBMS language constructs; better control of the development of database applications; and better quality of the overall design.

Acknowledgements

This work is supported by the Office of Naval Research and the Naval Command, Control and Ocean Surveillance Center RDT&E Division, and by the Rome Laboratories under contract No. F30602-93-C-0175. We are particularly thankful to Ms. Leah Wong for her constant cooperation and support. Dr.Ray Liuzzi of Rome Laboratories also provided valuable assistance in this project.

References

[1] C. Batini, S. Ceri, and S.B. Navathe. *Conceptual Database Design: an Entity-Relationship Approach.* Benjamin/Cummings, 1992.

[2] S. Chakravarthy and D. Mishra *Snoop: An Expressive Event Specification Language For Active Databases.* Technical Report UF-CIS-TR-93-007, University of Florida, March 1993.

[3] R. Elmasri and S.B. Navathe *Fundamentals of Database Systems.* Benjamin/Cummings, 1994.

[4] Asterio Kiyoshi Tanaka. *On Conceptual Design of Active Databases.* Ph. D Thesis, College of Computing, Georgia Institute of Technology, November 1992.

[5] Tansel A. U. et al. *Temporal Databases.* Benjamin/Cummings, 1994.

Appendix

User Manual

ER-GT is an X-Windows based Motif application that allows the user to specify the conceptual database schemas in terms of Entity and Relationships. On top of that, it allows the specification of the active behavior of the system in terms of events and rules. The resulting diagram is and ER^2 diagram which is an enhanced version of the ER diagram.

Startup

The program is located in *~active/ergt*. To run the program, enter *active_tool*. You will then see a window with a menu bar at the top.

Menus

The menu consists of the following items:

File Functions	Consist of the following submenus:
- Load Schema	Load an existing schema from disk.
- Add Schema	Add an existing schema from disk.
- Save Schema	Saves the current schema to disk.
- Quit Tool	Exit the application.
Edit Functions	Consist of the following submenus:
- Delete	Delete an object from the diagram.
- Move	Move an object in the diagram to a new location.
- Refresh	Redraws the diagram.
Add Functions	Consist of the following submenus:
- Entity	Add an Entity (represented by a rectangle).
- Relation	Add a Relation (represented by a diamond).
- Conn_Arc	Add a connecting arc between objects (represented by a line).
- Text	Add text to objects including the description and other parameters.
- Page	Add a new page.
- Event	Add an event (represented by a circle).
- Rule	Add a rule (represented by a parallelogram).

Next Page	Flip to the next page.
Prev Page	Flip to the previous page.
First Page	Flip to the first page.
Last Page	Flip to the last page.

Adding Objects

For adding an object to the diagram, select one of the menu options under the "Add Functions" menu. Then move the mouse to the drawing area. Press down the mouse button on this area. The object will be drawn at that location. You can draw as many objects of this type as you want. For instance, if you have chosen the Entity option, clicking the mouse button will draw rectangles on the drawing area. After this, you can choose another object, say relation. Now, the clicking action will draw diamonds on the drawing area. The same is true for both Events and Rules.

Adding Connecting Arcs

To connect the objects, choose the "Conn_Arc" option under the "Add Functions" menu. To draw the arc, position the mouse on the first object and press down the mouse button. Then, while holding the mouse button, drag the mouse to the second object. You will see the cursor change into a "pen" and a line will appear while you were dragging the mouse. Release the mouse button when the cursor is in the second object and an arc will be drawn between the two objects. Arcs can only be drawn between two objects that are not the same. Also, you must position the cursor on the first object before pressing the mouse button to start the drawing. Nothing will happen if the mouse if pressed outside an existing object. Moreover, you must end the mouse drag on the second object.

Adding Text

To add a text, choose the "Text" option under the "Add Functions" menu. Then position the cursor onto the object which you want text to be added and click on the mouse button. An appropriate dialog box will appear and the details of the objects can be added. For adding text to arcs, you have to position the cursor at the text description in the midpoint of the arc.

Deleting Objects

To delete objects on the drawing area, choose the "Delete" option under the "Edit Functions" menu. Then position the cursor at the object you want to delete. Click and the mouse

button and the object will be deleted. While in this delete mode, you may deleted as many objects as you like. Also, if there are connecting arcs connected to this object, the arcs will be deleted automatically.

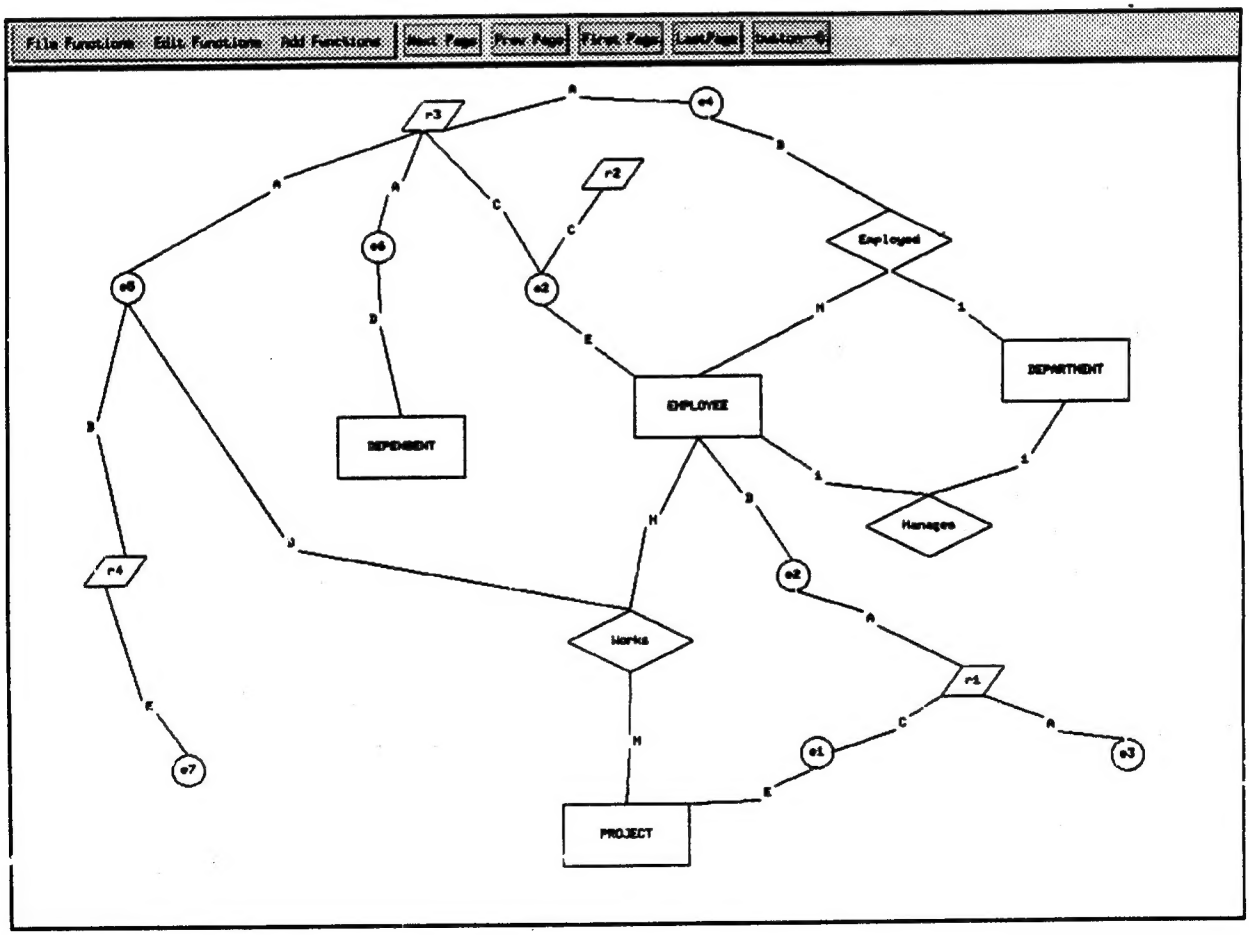
Moving Objects

To move objects on the drawing area, choose the "Move" option under the "Edit Functions" menu. Then position the cursor at the object you want to move. Press and the mouse button and hold it. Move the mouse by dragging the cursor to the new location. The object will move on the drawing area. Release the mouse button and the object will be placed at the new position. If there are connecting arcs to the object, the arcs will be adjusted accordingly.

Refresh the Drawing Area

To refresh the drawing area, choose the "Refresh" option under the "Edit Functions" menu. The drawing area and all the objects will be redrawn.

Screen Example



*U.S. GOVERNMENT PRINTING OFFICE: 1995-610-126-50177

Rome Laboratory

Customer Satisfaction Survey

RL-TR-_____

Please complete this survey, and mail to RL/IMPS, 26 Electronic Pky, Griffiss AFB NY 13441-4514. Your assessment and feedback regarding this technical report will allow Rome Laboratory to have a vehicle to continuously improve our methods of research, publication, and customer satisfaction. Your assistance is greatly appreciated.
Thank You

_____

_____

Organization Name:_____(Optional)

Organization POC: _____(Optional)

Address:_____

1. On a scale of 1 to 5 how would you rate the technology developed under this research?

5-Extremely Useful 1-Not Useful/Wasteful

Rating_____

Please use the space below to comment on your rating. Please suggest improvements. Use the back of this sheet if necessary.

2. Do any specific areas of the report stand out as exceptional?

Yes___ No_____

If yes, please identify the area(s), and comment on what aspects make them "stand out."

3. Do any specific areas of the report stand out as inferior?

Yes___ No___

If yes, please identify the area(s), and comment on what aspects make them "stand out."

4. Please utilize the space below to comment on any other aspects of the report. Comments on both technical content and reporting format are desired.